

Logo

Description automatically generated

**UniWow Footwear: Order Your Perfect Pair**

**POINT-OF-SALE SYSTEM**

IT 1 – Platform Technology Project

Code 6178

**SUBMITTED BY:**

Ariel July B. Traje

Ian Van Lawrence E. Castones

Lady May M. Peligro

Ralph Joseph B. Quiñones

**SUBMITTED TO:** Prof. Sheila Marie Villarin

**Program Information**

A point-of-sale (POS) system we developed using this 8086 program in assembly language was intended to be used by a cashier or an administrator. We used 4 of the 8 general-purpose registers in the ways listed below:

• AX: used to keep data temporarily, holding the value of the @DATA segment before it is transferred to the DS register;

• BX: a base register for memory operations;

• CX: used to count loops by the quantity of the pair of shoes; and

• DX: used to specify the memory address of the string to be displayed as a parameter to the "displayString" operation.

In our application, we also made use of multiple functions to perform some specific tasks, such as:

• MOV: a method for moving values across registers;

• PUSH: increases the stock's value;

• CALL: used to invoke a process that completes particular tasks;

• RET: when the process is finished, it is called to return to the main code; and more.

The program's flow is simple, considering only the cashier or store administrator will be able to use it. When the program is run, a table consisting of details about item numbers, brands, product names, prices, and sizes of various shoes will be displayed.

The process starts when the cashier or administrator enters the customer's desired pair of shoes' item number. After entering the item number, the system prompts the cashier to enter how many pairs of shoes the customer plans to purchase.

Afterwards, the algorithm multiplies the quantity by the fixed price to determine the total amount of shoes being purchased. The calculated amount is then shown, allowing the cashier or administrator to review the data and continue with the transaction if everything is correct.

With the help of this POS system, managing shoe sales is simplified and made more effective. It enables the cashier or administrator to swiftly input the item number and amount, giving the customer's purchase an exact representation. The functionality of this program makes sure that the total number of shoes is accurately computed, lowering the possibility of mistakes during the transaction.

**Source Code**

.MODEL LARGE

.STACK 1000H

.DATA

;DECLARED STRINGS

INTRO DB 10,13,10,13,' UNIWOW FOOTWEAR $',

INTRO1 DB 10,13,10,13,' LIMIT TO 2 PAIRS ONLY PER SHOES $',

INTRO2 DB 10,13,10,13,' UNISEX $',

line1 DB 10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$',

INFO DB 10,13,'|ITEM NO.| PRODUCTS | BRANDS | PRICE | SIZES | Discount |$',

line2 DB 10,13,'|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|$',

NIKE DB 10,13,'| 1 | Gaz it Zoom | NIKE | 2600 PHP | 35-45 | (1)290 (2)650 PHP |$',

CONVERSE DB 10,13,'| 2 | Ding Taylor | CONVERSE | 2525 PHP | 35-43 | (1)330 (2)700 PHP |$',

WORLDB DB 10,13,'| 3 | Yel it Baby | WORLDB | 2925 PHP | 35-43 | (1)400 (2)890 PHP |$',

ADIDAS DB 10,13,'| 4 | Wreck it Ralp | ADIDAS | 1675 PHP | 35-43 | (1)250 (2)650 PHP |$',

VANS DB 10,13,'| 5 | Old Skool | VANS | 1000 PHP | 35-43 | (1)170 (2)500 PHP |$',

line3 DB 10,13,'|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$',

ENTER DB 10,13,'ENTER ITEM NUMBER: $'

I\_QUANTITY DB 10,13,'QUANTITY OF SHOES: $'

AGAIN DB 10,13,10,13,'DO YOU WANT TO BUY MORE ITEMS? [1]YES [2]NO: $'

ER\_MSG DB 10,13,'IT SHOULD BE 1 TO 2 PAIRS ONLY $'

ER\_MSG1 DB 10,13,'ERROR THE INPUT IS NOT WITHIN THE PARAMETERS! PLEASE SELECT FROM 1 TO 5 $'

CHOICE DB 10,13,'ENTER YOUR CHOICE: $'

FT DB 10,13,'TOTAL AMOUNT: $'

ERR DB 0DH,0AH,' START FROM THE BEGINNING $'

ERR2 DB 0DH,0AH,'WRONG INPUT!!! PRESS 1 OR 2 $'

R DB 0DH,0AH,0DH,0AH,'TOTAL AMOUNT: $'

E\_DISCOUNT DB 10,13,'ENTER THE AMOUNT OF DISCOUNT: $'

ex DB 10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$'

ex1 DB 9,9,10,13,'| |$'

ex2 DB 9,9,10,13,'| THANK YOU FOR PURCHASING @ UNIWOW FOOTWEAR... PLEASE COME AGAIN!! |$'

ex3 DB 9,9,10,13,'| |$'

ex4 DB 10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$'

A DW ? ;DECLARED VARIABLES

B DW ?

C DW ?

S DW 0,'$'

NL DB 0DH,0AH,'$' ;NEW LINE

.CODE

MOV AX, @DATA

MOV DS, AX

LEA DX,INTRO ;PRINT INTRO STRING

CALL displayString

LEA DX, INTRO1

CALL displayString

LEA DX, INTRO2

CALL displayString

CALL newLine ;PRINT A NEW LINE

JMP BEGINTOP ;DIRECTLY GO TO BEGINTOP LEBEL WHERE USER WILL GIVE INPUT

ERROR121:

CALL newLine

LEA DX,ER\_MSG1 ;PRINT ERROR MESSAGE

CALL displayString

CALL newLine

;IF USER GIVES AN ERROR THEN USER WILL BE ASKED TO INPUT AGAIN

LEA DX,ERR

CALL displayString

BEGINTOP:

CALL newLine

LEA DX,line1 ;LINE 1

CALL displayString

CALL newLine

LEA DX,INFO ;PRINT INFO STRING

CALL displayString

LEA DX,line2 ;LINE 2

CALL displayString

CALL newLine

LEA DX,NIKE

CALL displayString

LEA DX,CONVERSE

CALL displayString

LEA DX,WORLDB

CALL displayString

LEA DX,ADIDAS

CALL displayString

LEA DX,VANS

CALL displayString

LEA DX,line3 ;LINE 3

CALL displayString

CALL newLine ;PRINT NEW LINE

LEA DX,ENTER ;PRINT ENTER STRING

CALL displayString

CALL input ;TAKE AN INPUT & SAVED TO AL

CMP AL,49 ;IF AL=1 GO TO NIKE LABEL

JE NIKE1

CMP AL,50 ;IF AL=2 GO TO CONVERSE LABEL

JE CONVERSE1

CMP AL,51 ;IF AL=3 GO TO WORLDB LABEL

JE WORLDB1

CMP AL,52 ;IF AL=4 GO TO ADIDAS LABEL

JE ADIDAS1

CMP AL,53 ;IF AL=5 GO TO VANS LABEL

JE VANS1

JMP ERROR121 ;ERROR ENTER AGAIN

displayString PROC

MOV AH,9 ;DISPLAY THE STRING

INT 21H

RET

displayString ENDP

input PROC ;CAN INPUT A STRING

MOV ah,01h

INT 21h

RET

input ENDP

newLine PROC

LEA DX,NL ;PRINT A NEW LINE

CALL displayString

RET

newLine ENDP

NIKE1:

MOV A,2600 ;INPUT THE PRICE OF NIKE WHICH IS 2600

JMP QUANTITY

CONVERSE1:

MOV A,2525 ;INPUT THE PRICE OF CONVERSE WHICH IS 2525

JMP QUANTITY

WORLDB1:

MOV A,2925 ;INPUT THE PRICE OF WORLDB WHICH IS 2925

JMP QUANTITY

ADIDAS1:

MOV A,1675 ;INPUT THE PRICE OF ADIDAS WHICH IS 1675

JMP QUANTITY

VANS1:

MOV A,1000 ;INPUT THE PRICE OF VANS WHICH IS 1000

JMP QUANTITY

;AFTER MOVING PRICE PROGRAM WILL JUMP TO QUANTITY LEBEL

QUANTITY:

CALL newLine

LEA DX,I\_QUANTITY ;PRINT ENTER QUANTITY STRING

CALL displayString

JMP MULTI ;PROGRAM WILL GO TO MULTI LABEL WHERE THE PRICE WILL BE MULTIPLIE WITH THE AMOUNT

ASK:

LEA DX,AGAIN ;PRINT AGAIN IF USER WANTS TO BUY MORE

CALL displayString

CALL input ;TAKES THE INPUT OF YES OR NO

CMP AL,49 ;IF YES, THEN AGAIN GO TO SHOPLIST MENU AND BUY AGAIN

JE BEGINTOP

CMP AL,50

JE OUTPUT2 ;IF NO, PROGRAM WILL GIVE THE TOTAL OUTPUT

LEA DX,ERR2

CALL displayString ;IF ANY WRONG INPUT, PRINT ERROR MESSAGE AND AGAIN ASK TO BUY AGAIN

JMP ASK

ERROR:

CALL newLine

LEA DX,ER\_MSG ;PRINT ERROR MESSAGE

CALL displayString

JMP QUANTITY ;JUMP TO QUANTITY LEBEL

ER\_DISCOUNT:

LEA DX,ER\_MSG ;DURING DISCOUNT INPUT IF WRONG INPUT IS PRESSES ERROR MESSSAGE WILL SHOW

CALL displayString

CALL newLine

JMP INPUT\_SUB ;DIRECLTY JUMP TO INPUT OF DISCOUNT

MULTI:

MOV BL,10 ;COLOR CODE

MOV AH,9

MOV AL,0

INT 10H

INDEC3 PROC ;INDEC3 IS FOR TAKING INPUT FOR MULTIPLY WITH THE GIVEN AMOUNT

PUSH BX ;TAKE VALUES INTO STACK

PUSH CX

PUSH DX

XOR BX,BX ;HOLDS TOTAL

XOR CX,CX ;SIGN

CALL input ;TAKE CHARACTER IN AL

REPEAT4:

CMP AL,48 ;IF AL<0, PRINT ERROR MESSAGE

JL ERROR

CMP AL,50 ;IF AL>5, PRINT ERROR MESSAGE

JG ERROR

AND AX,00FH ;CONVERT TO DIGIT

PUSH AX ;SAVE ON STACK

MOV AX,10 ;GET 10

MUL BX ;AX=TOTAL \* 10

POP BX ;GET DIGIT BACK

ADD BX,AX ;TOTAL = TOTAL X 10 +DIGIT

CALL input ;TAKE AN INPUT

CMP AL,0DH ;CARRIAGE RETURN

JNE REPEAT4 ;IF NO CARRIEGE RETURN THEN MOVE ON

MOV AX,BX ;STORE IN AX

JMP MUL\_

POP DX ;RESTORE REGISTERS

POP CX

POP BX

RET ;AND RETURN

INDEC3 ENDP ;END OF INDEC3

ADD\_:

;SECOND VALUE STORED IN B

MOV B,AX

XOR AX,AX ;CLEAR AX

MOV AX,B ;MOV B TO AX

ADD A,AX ;ADD A WITH AX

MOV AX,A ;MOV A TO AX

PUSH AX ;TAKE AX INTO STACK

JMP ExitP

SUB\_:

;SECOND VALUE STORED IN B

MOV B,AX

LEA DX,R ;PRINT PRESENT AMOUNT STRING

CALL displayString

XOR AX,AX ;CLEAR AX

MOV AX,B ;MOV B TO AX

SUB A,AX ;SUBSTRACT AX FROM A

MOV AX,A ;MOV A TO AX

PUSH AX

ADD S,AX

JMP OUTPUT

MUL\_:

;SECOND VALUE STORED IN B

MOV B,AX

LEA DX,E\_DISCOUNT ;PRINT ENTER DISCOUNT STRING

CALL displayString

XOR AX,AX ;CLEAR AX

MOV AX,B

MUL A ;MULTIPLY A WITH AX

PUSH AX ;TAKE AX INTO STACK

MOV A,AX

JMP INPUT\_SUB ;JUMP TO INP1UT\_SUB

JMP OUTPUT

INPUT\_ADD:

INDEC1 PROC ;INDEC PROC1 IS FOR ADDING THE PRESENT AMOUNTS INTO TOTAL

PUSH BX ;TAKE THE VALUES IN STACK

PUSH CX

PUSH DX

BEGIN1:

XOR BX,BX ;HOLDS TOTAL

XOR CX,CX ;SIGN

CALL input ;TAKE CHARACTER IN AL

REPEAT2:

;IF AL<0, PRINT ERROR MESSAGE

CMP AL,48

JL ERROR

CMP AL,50 ;IF AL>2, PRINT ERROR MESSAGE

JG ERROR

AND AX,00FH ;CONVERT TO DIGIT

PUSH AX ;SAVE ON STACK

MOV AX,10 ;GET 10

MUL BX ;AX=TOTAL \* 10

POP BX ;GET DIGIT BACK

ADD BX,AX ;TOTAL = TOTAL X 10 +DIGIT

CALL input ;TAKE VALUE INTO AL

CMP AL,0DH ;CARRIAGE RETURN

JNE REPEAT2 ;NO KEEP GOING

MOV AX,BX ;STORE IN AX

JMP ADD\_ ;JUMP TO ADD\_ TO STORE THE TOTAL VALUE

POP DX ;RESTORE REGISTERS

POP CX

POP BX

RET ;AND RETURN

INDEC1 ENDP

INPUT\_SUB:

INDEC2 PROC

PUSH BX ;SAVE TO STACK

PUSH CX

PUSH DX

XOR BX,BX ;HOLDS TOTAL

XOR CX,CX ;SIGN

CALL input ;CHAR IN AL

REPEAT3:

CMP AL,48 ;IF AL<0, PRINT ERROR MESSAGE

JL ER\_DISCOUNT

CMP AL,57 ;IF AL>9, PRINT ERROR MESSAGE

JG ER\_DISCOUNT

AND AX,00FH ;CONVERT TO DIGIT

PUSH AX ;SAVE ON STACK

MOV AX,10 ;GET 10

MUL BX ;AX=TOTAL \* 10

POP BX ;GET DIGIT BACK

ADD BX,AX ;TOTAL = TOTAL X 10 +DIGIT

MOV AH,1

INT 21H

CMP AL,0DH ;CARRIAGE RETURN

JNE REPEAT3 ;NO KEEP GOING

MOV AX,BX ;STORE IN AX

OR CX,CX ;NEG NUM

JMP SUB\_

POP DX ;RESTORE REGISTERS

POP CX

POP BX

RET ;AND RETURN

INDEC2 ENDP

OUTPUT:

;OUTDEC PROC IS FOR GIVING THE OUTPUT OF THE PRESENT AMOUNT

OUTDEC PROC

PUSH AX ;SAVE REGISTERS

PUSH BX

PUSH CX

PUSH DX

XOR CX,CX ;CX COUNTS DIGITS

MOV BX,10D ;BX HAS DIVISOR

REPEAT1:

XOR DX,DX ;PREP HIGH WORD

DIV BX ;AX = QUOTIENT, DX=REMAINDER

PUSH DX ;SAVE REMAINDER ON STACK

INC CX ;COUNT = COUNT +1

OR AX,AX ;QUOTIENT = 0?

JNE REPEAT1 ;NO, KEEP GOING

MOV AH,2 ;PRINT CHAR FUNCTION

PRINT\_LOOP:

POP DX ;DIGIT IN DL

OR DL,30H ;CONVERT TO CHAR

INT 21H ;PRINT DIGIT

LOOP PRINT\_LOOP ;LOOP UNTILL DONE

POP DX

POP CX ;RESTORE REGISTERS

POP BX

POP AX

JMP ASK

RET

OUTDEC ENDP

OUTPUT2:

LEA DX,FT ;PRINT FINAL TOTAL

CALL displayString

XOR AX,AX ;CLEAR AX

MOV AX,S ;SET AX INTO 0

;OUTDEC2 IS FOR GIVING THE TOTAL OUTPUT OF THE AMOUNT

OUTDEC2 PROC

PUSH AX ;SAVE REGISTERS

PUSH BX

PUSH CX

PUSH DX

XOR CX,CX ;CX COUNTS DIGITS

MOV BX,10D ;BX HAS DIVISOR

REPEAT12:

XOR DX,DX ;PREP HIGH WORD

DIV BX ;AX = QUOTIENT, DX=REMAINDER

PUSH DX ;SAVE REMAINDER ON STACK

INC CX ;COUNT = COUNT +1

OR AX,AX ;QUOTIENT = 0?

JNE REPEAT12 ;NO, KEEP GOING

MOV AH,2 ;PRINT CHAR FUNCTION

PRINT\_LOOP2:

POP DX ;DIGIT IN DL

OR DL,30H ;CONVERT TO CHAR

INT 21H ;PRINT DIGIT

LOOP PRINT\_LOOP2 ;LOOP UNTILL DONE

POP DX

POP CX ;RESTORE REGISTERS

POP BX

POP AX

OUTDEC2 ENDP

ExitP:

; Thanks note

CALL input

LEA DX,ex

CALL displayString

LEA DX,ex1

CALL displayString ;Calling the method displayString

LEA DX,ex2

CALL displayString

LEA DX,ex3

CALL displayString

LEA DX,ex4

CALL displayString

.EXIT

ENDP

ret

**Screenshots**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated